



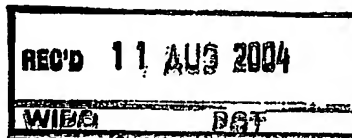
Europäisches
Patentamt

European
Patent Office

Office européen
des brevets

PCT/IB04/51420

THDE030280 EP



Bescheinigung

Certificate

Attestation

Die angehefteten Unterla-
gen stimmen mit der
ursprünglich eingereichten
Fassung der auf dem näch-
sten Blatt bezeichneten
europäischen Patentanmel-
dung überein.

The attached documents
are exact copies of the
European patent application
described on the following
page, as originally filed.

Les documents fixés à
cette attestation sont
conformes à la version
initialement déposée de
la demande de brevet
européen spécifiée à la
page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

03102501.8

**PRIORITY
DOCUMENT**
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH RULE 17.1(a) OR (b)

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk



Anmeldung Nr:
Application no.: 03102501.8
Demande no:

Anmeldetag:
Date of filing: 12.08.03
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

Philips Intellectual Property & Standards
GmbH
Steindamm 94
20099 Hamburg
ALLEMAGNE
Koninklijke Philips Electronics N.V.
Groenewoudseweg 1
5621 BA Eindhoven
PAYS-BAS

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se référer à la description.)

Spracheingabeschnittstelle für Dialogsysteme

In Anspruch genommene Priorität(en) / Priority(ies) claimed / Priorité(s)
revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

G10L/

Am Anmeldetag benannte Vertragsstaaten/Contracting states designated at date of
filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LU MC NL
PT RO SE SI SK TR LI

BESCHREIBUNG

Spracheingabeschnittstelle für Dialogsysteme

Die Erfindung betrifft ein Verfahren zum Betrieb eines Dialogsystems mit einer Sprach-
5 eingabeschnittstelle. Sie betrifft außerdem ein Verfahren und ein System zur Erstellung einer
Spracheingabeschnittstelle, eine entsprechende Spracheingabe-Schnittstelle und ein
Dialogsystem mit einer derartigen Spracheingabeschnittstelle.

10 Sprachgesteuerte Dialogsysteme haben ein breites gewerbliches Anwendungsspektrum. Sie
werden in Sprachportalen aller Art eingesetzt, beispielsweise beim Telefon-Banking, der
sprachgesteuerten automatischen Warenausgabe, zur Sprachsteuerung von Freisprechanlagen
in Fahrzeugen oder in sogenannten Home-Dialog-Systemen. Darüber hinaus ist es möglich,
diese Technologie in automatische Übersetzungs- und Diktatsysteme einzusetzen.

15 Bei der Entwicklung und Herstellung von Sprachdialogsystemen besteht das allgemeine
Problem, die Spracheingabe des Benutzers eines Dialogsystems zuverlässig zu erkennen, auf
effiziente Art und Weise zu verarbeiten und in die vom Benutzer gewünschte systeminterne
Reaktion umzusetzen. Je nach Größe des Systems und Komplexität des zu beherrschenden
Dialogs ergeben sich hierbei vielfältige einander bedingende Teilprobleme: Die
20 Spracherkennung gliedert sich in der Regel in einen syntaktischen Teilschritt, der eine gültige
Äußerung detektiert, und einen semantischen Teilschritt, der die als gültig erkannte Äußerung
auf ihre vom System verwendbare Bedeutung abbildet. Diese Spracherkennung erfolgt daher
meist durch eine spezialisierte Sprachverarbeitungsschnittstelle des Dialogsystems, die die
Äußerung des Benutzers beispielsweise durch ein Mikrophon aufnimmt, sie in ein digitales
25 Sprachsignal umsetzt und anschließend die Spracherkennung durchführt.

Die Verarbeitung des digitalen Sprachsignals durch die Spracherkennung wird zu einem großen Teil von Softwarekomponenten durchgeführt. In der Regel liegt daher als Ergebnis der Spracherkennung die Bedeutung einer Äußerung in Form von Daten und/oder

5 Programmanweisungen vor. Diese Programmanweisungen werden schließlich ausgeführt bzw. die Daten verwendet und führen somit zu der von Benutzer beabsichtigten Reaktion des Dialogsystems. Diese Reaktion kann beispielsweise aus einer elektronischen oder mechanischen Handlung (z.B. Auswerfen von Geldscheinen bei einem sprachgesteuerten Geldautomaten) oder aus einer rein programmbezogenen und somit für den Benutzer zunächst

10 transparenten Datenmanipulation bestehen (z.B. Änderung des Kontostandes). Üblicherweise wird daher die tatsächliche Umsetzung der Bedeutung einer Sprachäußerung, also das Ausführen der "semantischen" Programmanweisungen durch eine von der Spracheingabeschnittstelle logisch getrennte Applikation, beispielsweise ein Steuerprogramm, durchgeführt. Die Steuerung des Dialogsystems selbst erfolgt in der Regel durch einen

15 sogenannten Dialogmanager auf Basis einer vorgegebenen deterministischen Dialogbeschreibung.

Abhängig von dem Stadium des Dialogs zwischen Benutzer und Dialogsystem befindet sich das Dialogsystem zu einer bestimmten Zeit in einem definierten Zustand (spezifiziert durch die

20 Dialogbeschreibung) und geht bei einer gültigen Anweisung des Benutzers in einen entsprechend anderen Zustand über. Für jeden dieser Zustandsübergänge muss die Spracheingabeschnittstelle eine individuelle Spracherkennung vornehmen, da bei jedem Zustandsübergang andere Äußerungen erkannt und eindeutig auf die korrekte Semantik abgebildet werden müssen. So wird beispielsweise in einem Zustand lediglich eine Bestätigung

25 durch ein "Ja" erwartet, während in einem anderen Fall einer komplexen Äußerung dezidierte Informationen (z.B. eine Kontonummer) entnommen werden müssen. In der Praxis müssen bei jedem Zustandsübergang mehrere synonyme Äußerungen auf den gleichen semantischen Sinn abgebildet werden, so verfolgen z.B. die Anweisungen "Halt", "Stop", "Ende" und "Schluss" alle das gleiche Ziel, nämlich einen Prozess zu beenden.

Zur Behandlung der Komplexität des Problems des Verstehens und der Weiterverarbeitung einer Sprachäußerung gibt es unterschiedliche Ansätze. So ist es prinzipiell möglich, für jede gültige Äußerung eines jeden Zustandsübergangs ein prototypisches Sprachsignal

- 5 abzuspeichern, mit denen eine konkrete Äußerung silben- oder wortweise mit stochastischen oder spektralen Methoden verglichen werden muss. Eine adäquate Reaktion auf eine Sprachäußerung kann programmiertechnisch als direkte Folge auf das Erkennen einer bestimmten Äußerung realisiert werden. Bei komplexen Dialogen, bei denen es unter Umständen nötig ist, detaillierte Informationen zu übergeben, führt dieser starre Ansatz zu der
- 10 Notwendigkeit, einerseits alle erlaubten synonymen Varianten einer Äußerung vorliegen zu haben, um diese bei Bedarf mit der Äußerung eines Benutzers zu vergleichen, und andererseits benutzerspezifische Information durch spezielle Programmroutinen weiterzuverarbeiten. Es liegt auf der Hand, dass diese Lösung inflexibel und durch einen Betreiber eines Dialogsystems nur sehr schwer erweiterbar und adaptierbar ist.

15

Einen andere Strategie verfolgt der dynamischere grammatikalische Ansatz, der zur Spracherkennung linguistisch-grammatikalische Modelle in Form von formalen Grammatiken einsetzt. Formale Grammatiken sind algebraische Strukturen, die aus Ersetzungsregeln, Terminalwörtern, Nichtterminalwörtern und einem Startwort bestehen. Die Ersetzungsregeln

20 geben Vorschriften, nach denen Nichtterminalwörter strukturell in Wortketten bestehend aus Nichtterminal- und Terminalwörtern überführt (abgeleitet) werden können. Alle Sätze, die nur aus Terminalwörtern bestehen und durch Anwendung der Ersetzungsregeln ausgehend von dem Startwort erzeugt werden können, repräsentieren gültige Sätze der durch die formalen Grammatik spezifizierten Sprache.

25

Beim grammatikalischen Ansatz werden für jeden Zustandsübergang eines Dialogsystems durch die Ersetzungsregeln einer formalen Grammatik die erlaubten Satzstrukturen generisch vorgegeben und durch die Terminalwörter das Vokabular der Sprache spezifiziert, deren sämtliche Sätze als gültige Äußerung eines Benutzers akzeptiert werden. Eine konkrete

Sprachäußerung wird also durch die Überprüfung verifiziert, ob sie durch Anwenden der Ersetzungsregeln und Einsetzen des Vokabulars aus dem Startwort der entsprechenden formalen Grammatik ableitbar ist. Möglich sind auch Ausführungen, bei denen nur die sinntragenden Wörter an den durch die Ersetzungsregel gegebenen Stellen der Satzstruktur
5 überprüft werden.

Neben dieser syntaktischen Verifikation eines Satzes muss die Spracherkennung jedem gültigen Satz seine Semantik, d.h. eine Bedeutung zuordnen, die in eine Reaktion des Systems umsetzbar ist. Die Semantik besteht aus Programmanweisungen und/oder Daten, die durch die
10 Applikation des Dialogsystems anwendbar sind. Zur Zuordnung von ausführbaren Programmanweisungen zu den entsprechenden syntaktischen Elementen werden häufig solche Grammatiken verwendet, die die Semantik in Form eines sogenannten Attributs mit dem zugehörigen Terminal-/Nichtterminalwort verknüpfen. Bei sogenannten synthetisierten Attributen wird dabei für Nichtterminalwörter der Attributwert aus den Attributen letztlich der
15 Terminalwörter berechnet. Bei sogenannten vererbten Attributen können zur Attributbe- rechnung auch Informationen aus dem übergeordneten Nichtterminal benutzt werden. Die Semantik einer Sprachäußerung wird dadurch bei der Ableitung des Satzes aus dem Startwort als Attribut oder Attributfolge implizit miterzeugt. Dadurch wird zumindest formal die direkte Abbildung der Syntax auf die Semantik ermöglicht.

20

Die US 6,434,529 B1 offenbart ein System, das sich einer objektorientierten Programm- technologie bedient und gültige Sprachäußerungen mittels formaler Grammatiken identifiziert. Die formale Grammatik und ihre Überprüfung ist in diesem System mittels einer Interpreter- sprache implementiert. Da zur semantischen Umsetzung der als syntaktisch korrekt erkannten
25 Satzelemente objektorientierte Klassen in einem übersetzten (kompilierten) Applikations- programm instanziiert bzw. ihre Methoden ausgeführt werden, ist eine Schnittstelle zwischen der von einem Interpreter durchzuführenden Syntaxanalyse und der semantischen Umsetzung in dem lauffähigen maschinensprachlichen Applikationsprogramm vorgesehen.

Diese Schnittstelle ist wie folgt realisiert: In der Spezifikation der Grammatik bzw. ihren Ersetzungsregeln sind den Terminal- und/oder den Nichtterminalwörtern semantische Attribute in Form von skriptsprachlichen Programmfragmenten zugeordnet. Diese semantischen Skriptfragmente werden während der syntaktischen Ableitung (dem sogenannten Parsing) der

- 5 Sprachäußerung gemäß der Anwendungsreihenfolge der Ersetzungsregeln in eine hierarchische Datenstruktur umgesetzt, die den gesprochenen Satz syntaxstrukturell repräsentiert. Anschließend wird die hierarchische Datenstruktur durch einen weiteren Parsing-Schritt in eine Tabelle umgesetzt und stellt schließlich eine vollständige, linear ausführbare, programmiersprachliche Repräsentation der Semantik der entsprechenden Äußerung dar, die aus skriptsprachlichen
- 10 Anweisungen zur Instanziierung eines Objektes oder zur Ausführung einer Methode im Applikationsprogramm besteht. Diese Repräsentation kann nun durch einen Parser/Interpreter ausgewertet werden, indem die entsprechenden Objekte direkt im Applikationsprogramm angelegt werden und die entsprechenden Methoden von diesem ausgeführt werden.

- 15 Die Nachteile dieser Technologie sind teilweise bereits anhand ihrer Beschreibung ersichtlich. Die Verwendung einer (u.U. proprietären) Interpretersprache zur Syntaxanalyse und einer Übersetzersprache für das Applikationsprogramm erfordert eine komplexe und komplizierte Schnittstelle zwischen der Spracheingabeschnittstelle und der Applikation, die zwei völlig unterschiedlichen Programmertechnologien repräsentieren.

20

Außerdem ist eine Erweiterung oder Änderung sowohl der grammatikalischen Sprachspezifikation als auch der Semantik-Skripte für einen Benutzer nicht ohne weiteres möglich, da er dazu zunächst die spezielle Skriptsprache erlernen muss. Darüber hinaus muss unter Umständen eine Erweiterung oder Änderung der Semantik durch Anpassen der entsprechenden

- 25 semantischer Programmfragmente auch in der Applikation implementiert und übersetzt (kompiliert) werden. Deshalb ist bei dieser Technologie die Sprache nicht während der Laufzeit des Dialogsystems veränderbar oder adaptierbar. Da bei der Umsetzung der Syntax in Semantik

(also zur Laufzeit des Dialogsystems) Parser bzw. Interpreter eingesetzt werden, entsteht zudem bei der Pflege der verschiedenen Komponenten des Systems ein erhöhter Wartungsaufwand.

5

Es ist eine Aufgabe der vorliegenden Erfindung, den Betrieb und Aufbau einer Spracheingabeschnittstelle eines Dialogsystems so zu ermöglichen, dass die zu erkennende Sprache durch eine einfache, schnelle und insbesondere leicht veränderbare Spezifikation einer formalen Grammatik definiert werden kann und Sprachäußerungen effizient auf ihre Semantik
10 abgebildet werden.

Diese Aufgabe wird zum einen durch ein Verfahren zum Betrieb eines Dialogsystems mit einer Spracheingabeschnittstelle und einer mit der Spracheingabeschnittstelle zusammenwirkenden Applikation gelöst, bei dem die Spracheingabeschnittstelle Audiosprachsignale eines Benutzers erfasst und direkt in ein Erkennungsergebnis in Form von Binärdaten umsetzt und dieses
15 Ergebnis der Applikation zur Ausführung bereitstellt. Unter Binärdaten sollen im folgenden Daten und/oder Programmanweisungen (oder Referenzen bzw. Zeiger darauf) verstanden werden, die ohne weitere Transformation oder Interpretation unmittelbar von der Applikation verwendbar/ausführbar sind, wobei die unmittelbar ausführbaren Daten von einem maschinensprachlichen Teilprogramm der Spracheingabeschnittstelle erzeugt werden. Hierunter ist
20 insbesondere auch der Fall zu verstehen, dass als Erkennungsergebnis ein oder mehrere maschinensprachliche Programmmodule erzeugt werden, die der Applikation zur unmittelbaren Ausführung bereitgestellt werden. Zum anderen wird die Aufgabe durch ein Verfahren zur Erstellung einer Spracheingabeschnittstelle für ein Dialogsystem mit einer mit der Spracheingabeschnittstelle zusammenwirkenden Applikation gelöst, welche die folgenden Schritte umfasst: Spezifizieren gültiger Spracheingabesignale durch eine formale Grammatik, wobei das gültige Vokabular der Spracheingabesignale als Terminalwörter der Grammatik definiert wird; Bereitstellen von Binärdaten, die die Semantik gültiger Audiosprachsignale repräsentieren und
25 die aus zur Systemlaufzeit durch die Applikation unmittelbar verwendbaren und von einem

Programmteil der Spracheingabeschnittstelle erzeugten Datenstrukturen bestehen oder aus unmittelbar von der Applikation ausführbaren Programmmodulen bestehen, und/oder Bereitstellen von Programmteilen, welche die Binärdaten erzeugen; Zuordnen der Binärdaten und/oder Programmteile zu einzelnen oder Kombinationen von Terminalwörtern oder

5 Nichtterminalwörtern, zur Abbildung eines gültigen Audiosprachsignals auf eine passende Semantik; Übersetzen der Programmteile und/oder der Programmmodule in Maschinensprache, derart, dass die übersetzten Programmteile beim Betrieb des Dialogsystems unmittelbar von der Applikation nutzbare Datenstrukturen erzeugen bzw. die übersetzten Programmmodule beim Betrieb des Dialogsystems unmittelbar von der Applikation ausführbar

10 sind, wobei die Datenstrukturen/Programmmodule die Semantik einer Sprachäußerung darstellen.

Erfindungsgemäß wird folglich die in ein Audiosignal umgewandelte Sprachäußerung des Benutzers von der Spracheingabeschnittstelle des Dialogsystems direkt in Binärdaten

15 transformiert, die die semantische Umsetzung der Spracheingabe und insofern das Erkennungsergebnis repräsentieren. Dieses Erkennungsergebnis kann unmittelbar von dem mit der Spracheingabeschnittstelle zusammenwirkenden Applikationsprogramm verwendet werden. Dass diese Binärdaten insbesondere auch aus einem oder mehrere maschinensprachlichen Programmmodule bestehen können, die unmittelbar von der Applikation ausführbar sind, wird

20 beispielsweise dadurch erreicht, dass die Spracheingabeschnittstelle in einer Übersetzersprache geschrieben ist und die Programmmodule des Erkennungsergebnisses, ebenfalls in einer -möglicherweise unterschiedlichen- Übersetzersprache implementiert wurden. Vorzugsweise sind diese Programmmodule in der gleichen Sprache geschrieben, in der auch die Spracherkennungslogik implementiert wurde. Sie können aber auch in einer Sprache geschrie-

25 ben und kompiliert sein, die auf der gleichen Plattform wie die Spracheingabeschnittstelle arbeitet. Dadurch ist es je nach verwendeter Übersetzersprache möglich, entweder die ausführbaren Programmmodule als solche oder Referenzen bzw. Zeiger auf diese Module dem Applikationsprogramm als Erkennungsergebnis zur unmittelbaren Ausführung bereitzustellen.

Besonders vorteilhaft ist dabei der Einsatz einer objektorientierten Programmiersprache, da hiermit einerseits die Programmmodule der Applikation in Form von Objekten bzw. Methoden von Objekten zur unmittelbaren Ausführung bereitgestellt werden können, und andererseits auch die von der Applikation unmittelbar zu verwendenden Datenstrukturen als Objekte einer objektorientierten Programmiersprache repräsentiert werden können.

Diese Erfindung bietet vielfältige Vorteile. Durch die Realisierung der Spracherkennung der Spracheingabeschnittstelle, insbesondere der Semantiksyntax, als ein von Prozessor unmittelbar ausführbares Maschinenprogramm (im Gegensatz zu einem nur mittelbar von einem Interpreter ausführbares Skriptprogramm) wird die direkte Generierung eines von dem maschinensprachlichen Applikationsprogramm unmittelbar verwendbaren Erkennungsergebnisses ermöglicht. Dadurch ergibt sich die größtmögliche Effizienz bei der Umsetzung der Sprachäußerung in eine adäquate Reaktion des Dialogsystems. Insbesondere wird dadurch eine komplizierte und datentechnisch aufwendige Abbildung der von einem skriptsprachlichen Parser aus einer formalen Grammatik gewonnenen semantischen Attribute bzw. Skriptprogrammfragmente in eine maschinensprachliche Repräsentation überflüssig. Weitere Vorteile ergeben sich aus der Möglichkeit, beim Aufbau oder der Spezifikation einer Spracheingabeschnittstelle durch einen Dienstleister oder bei ihrer Anpassung an neue Fakten (z.B. Sonderangebote bei einem Verkaufsautomat) herkömmliche Programmiersprachen wie C, C++, C# oder Java anstelle von proprietären Skriptsprachen des Herstellers der Spracheingabeschnittstelle verwenden zu können. Derartige Sprachen sind einem breiten Anwenderspektrum zumindest insoweit bekannt, dass einfache Anpassungen oder Erweiterung der Syntax der vom System zu verstehenden Sprachäußerungen oder der zugehörigen semantischen Programmmodule über eine entsprechende Eingabeschnittstelle oft ohne weiteres durchgeführt werden können. Es entfällt also die Notwendigkeit, zur Neukonfiguration oder zum Update des Dialogsystems eine proprietäre Sprache erlernen zu müssen. Für den Hersteller ergibt sich aus der Verwendung einer Übersetzersprache weiterhin der Vorteil einer einfacheren und

somit kostengünstigeren softwaretechnischen Wartung des Systems, da handelsübliche Standard-Compiler eingesetzt werden können und das Warten bzw. Weiterentwickeln einer eigenen Skriptsprache sowie der entsprechenden Parser und Interpreter entfällt.

- 5 Das Umsetzen der Sprachäußerung in semantische Programmmodule kann im einfachsten Falle durch eine direkte und eindeutige Zuordnung der möglichen Sprachäußerungen zu den entsprechenden Programmmodulen geschehen. Eine flexiblere, erweiterbare und effiziente Spracherkennung erhält man jedoch durch die methodische Trennung der Spracherkennung in einen Syntaxanalyse-Schritt und einen Semantiksyntax-Schritt. Durch die Definition der von
- 10 der Spracheingabeschnittstelle zu verstehenden Sprache mittels einer formalen Grammatik wird die Syntaxanalyse, also die Überprüfung einer Sprachäußerung, auf ihre Gültigkeit formalisiert und von der semantischen Umsetzung getrennt. Das gültige Vokabular der Sprache ergibt sich aus den Terminalwörtern der Grammatik, während die Satzstruktur durch die Ersetzungsregeln und die Nichtterminalwörter bestimmt wird. Da sowohl die Syntaxanalyse als
- 15 auch die Semantiksyntax von einem oder mehreren Maschinenprogrammen durchgeführt werden, wird das Erkennungsergebnis einer Sprachäußerung direkt in Form von Binärdaten, insbesondere Programmmodulen, erzeugt, die unmittelbar von der Applikation verwendbar/ ausführbar sind.. Beispiele dafür sind ein vom Prozessor linear abarbeitbares Programmmodul, das sich bei der Zuordnung eines semantischen, maschinensprachlichen Programmfragments
- 20 zu jedem Terminal- und Nichtterminalwort durch eine attributierte Grammatik, aus der Traversierung des Ableitungsbaums einer gültigen Sprachäußerung ergibt. Ein anderes Beispiel wäre eine binäre Datenstruktur, die einen Zeitpunkt beschreibt und als Attribut einer Zeitgrammatik aus deren Bestandteilen synthetisiert worden ist.
- 25 In vielen Fällen wird die Grammatik vor der Inbetriebnahme des Dialogsystems vollständig definiert und bleibt während des Betriebs unverändert. Vorzugsweise ist jedoch eine dynamische Änderung der Grammatik während des Betriebs des Dialogsystems möglich, indem die Syntax und Semantik der vom Dialogsystem zu verstehenden Sprache der Applikation beispielsweise in Form einer dynamisch bindbaren Bibliothek (dynamic linked library) zur

Verfügung gestellt werden. Dies ist bei häufiger Änderung von Sprachelementen oder semantischen Änderungen, beispielsweise bei Sonderangeboten oder wechselnden Informationsangeboten von großem Vorteil.

- 5 Besonders bevorzugt wird die Spracherkennung in einer objektorientierten Übersetzersprache implementiert. Das eröffnet eine effiziente und vom jeweiligen Anwender leicht zu modifizierende Realisierung von generischen Standard-Ersetzungsregeln von formalen Sprachen - wie z.B. einer Terminalregel, einer Kettenregel und einer Alternativregel - als objektorientierte Grammatik-Klassen. Die gemeinsamen Eigenschaften und Funktionalitäten, insbesondere eine
- 10 generische Parsing-Methode, dieser Grammatik-Klassen können beispielsweise von einer oder gar mehreren unspezifischeren Basisklassen geerbt werden. Ebenso können die Basis-klassen virtuelle Methoden durch Vererbung an die Grammatik-Klassen weitergeben, die bei Bedarf überladen bzw. überschrieben werden können, um konkrete Funktionalitäten wie beispielsweise bestimmte Parsing-Methoden zu realisieren. Mit den entsprechenden in den
- 15 jeweiligen Klassendefinitionen vorgesehenen Konstruktoren kann dann die Grammatik einer konkreten Sprache durch Instanziierung der generischen Grammatik-Klassen spezifiziert werden. Hierbei werden durch die Definition der Terminal- und Nichtterminalwörter konkrete Ersetzungsregeln als programmiersprachliche Objekte erzeugt. Jedes dieser Grammatik-Ob-
jekte besitzt dann eine individuelle Evaluierungs- bzw. Parsingmethode, die überprüft, ob die
- 20 entsprechende Regel auf die detektierte Phrase anwendbar ist. Die geeignete Anwendung der Ersetzungsregeln und damit die Gültigkeitsüberprüfung des gesamten Sprachsignals bzw. die Detektion der entsprechenden Phrase wird durch den Syntaxanalyse-Schritt der Spracherkennung gesteuert.
- 25 Durch die konsequente Umsetzung des systematisierenden Konzepts der vorliegenden Erfindung wird in einer bevorzugten Ausführungsform die methodische Trennung zwischen Syntaxanalyse und Semantiksynthese beibehalten, während die zeitliche Trennung ihrer Anwendung zum Zwecke der Effizienzsteigerung und Reduktion der Antwortzeiten zumindest teilweise

- aufgehoben wird. Bei der Verwendung attributierter Grammatiken werden während der Ableitung eines zu erkennenden Sprachsignals aus dem Startwort die entsprechenden semantischen Binärdaten (das Attribut) einer anwendbaren Ersetzungsregel direkt erzeugt. Sobald also zum Beispiel in einer Regel <„Viertel vor“ <Zahl von 1 bis 12>> die Zahl als
- 5 Ergebnis der Regel <Zahl von 1 bis 12> bekannt ist, kann eine entsprechende Uhrzeitdatenstruktur, in diesem Fall mit Wert „11:45 Uhr“, erzeugt werden. Falls andererseits bei der weiteren Anwendungen geeigneter Ersetzungsregeln die zur Ausführung eines semantischen Programmmoduls erforderlichen Parameter bekannt werden, kann dieses Programmmodul direkt von der Spracheingabeschnittstelle ausgeführt werden. Die Semantik wird also nicht
- 10 erst vollständig aus dem Sprachsignal extrahiert, sondern sie wird quasi-parallel während der syntaktischen Überprüfung bereits umgesetzt und ausgeführt. Statt Referenzen auf ausführbare Programmfragmente und entsprechende Parameter, übergibt die Spracheingabeschnittstelle dem Applikationsprogramm direkt die – andernfalls von der Applikation zu berechnenden – Ergebnisse. Diese besonders vorteilhafte Ausführungsform wird durch Realisierung der syn-
- 15 taktischen Überprüfung zur Spracherkennung, der semantischen Programmmodule und des Applikationsprogramms als maschinensprachliche Programme ermöglicht, weil die Programmeinheiten des Dialogsystems dadurch über geeignete Schnittstellen effizient kommunizieren und Daten austauschen können.
- 20 Bei einem objektorientierten Aufbau der Spracheingabeschnittstelle unter Verwendung attributierter Grammatiken können die semantischen Programmmodule als programmiersprachliche Objekte bzw. Methoden von Objekten realisiert werden. Diese zusätzliche Systematisierung der semantischen Seite wird von der vorliegenden Erfindung unterstützt, indem die Grammatik-Klassen so instanziiert werden können, dass sie anstelle der Standard-
- 25 werte (also beispielsweise einzelnen oder Listen von erkannten Terminal- und Nichtterminalworten) "semantische" Objekte zurückgeben, die durch das Überschreiben von virtuellen Methoden der jeweiligen Grammatik-Klassen definiert werden. Ebenso können bei Anwendung entsprechender Ersetzungsregeln (also beim Parsen des Sprachsignals) semantische

Objekte zurückgegeben werden, die aus den beim Parsen zurückgegebenen Werten berechnet werden.

Das oben genannte erfindungsgemäße Verfahren zur Erstellung einer Spracheingabeschnitt-
5 stelle bietet die Möglichkeit einer einfachen, schnellen und fehlerarmen Neuerstellung bzw.
Konfiguration von Sprachverarbeitungsschnittstellen. Dabei wird zur Spezifikation der zu
erkennenden Sprache zunächst eine formale Grammatik generisch definiert, indem das gültige
Vokabular der Sprache durch die Terminalwörter und die gültige Struktur der Sprachäuße-
rungen durch die Ersetzungsregeln bzw. die Nichtterminalwörter bestimmt wird. Nach der
10 Spezifikation dieser syntaktischen Ebene wird die semantische Ebene durch die Bereitstellung
von in einer Übersetzersprache geschriebenen Programmmodulen spezifiziert, deren maschi-
nensprachlichen Übersetzungen zur Laufzeit des Dialogsystems zur Abbildung der syntak-
tischen Struktur auf die entsprechende Semantik einer Sprachäußerung geeignet kombiniert
werden können; weiter können Binärdaten spezifiziert werden, und/oder Programmteile, die
15 die Binärdaten und/oder die Programmmodule zur Laufzeit geeignet kombinieren. Zwischen
der syntaktischen und der semantischen Ebene wird eine eindeutige Zuordnung definiert, so
dass jedem Terminal- und Nichtterminalwort ein seine Semantik beschreibendes Programm-
modul zugeordnet wird. Da die semantischen Programmmodule in einer Übersetzersprache
(z.B. C, C++ o.Ä.) implementiert werden, müssen sie nach ihrer Definition mit dem entspre-
20 chenden Compiler übersetzt werden, um sie für die unmittelbare Ausführung beim Betrieb des
Dialogsystems bereitzustellen.

Dieses Verfahren besitzt mehrere Vorteile. Es ermöglicht einerseits einem Dienstleister, der
Spracheingabeschnittstellen für bestimmte Anwendungen entwirft oder konfiguriert, auf sehr
25 einfache Weise Syntax und Semantik mittels einer bekannten Übersetzersprache zu spezifi-
zieren. Es muss also nicht eine mitunter komplexe proprietäre (Skript-) Sprache des Her-
stellers erlernt werden. Darüber hinaus ist die Verwendung einer Übersetzersprache aufgrund
der Überprüfung durch den Übersetzer und der Manipulationssicherheit der Maschinenpro-
gramme weniger fehleranfällig und beim Endkunden stabiler und schneller einsetzbar.

Die übersetzten semantischen Programmmodule können nach der Spezifikation der Semantik dem Dialogsystem eines Endkunden beispielsweise als dynamische oder statische Bibliotheken zur Verfügung gestellt werden. Im Falle einer dynamisch zu bindenden Bibliothek muss das

5 Applikationsprogramm des Dialogsystems nach der Bereitstellung geänderter semantischer Programmmodule nicht neu übersetzt werden, da es die auszuführenden Programmmodule über Referenzierung ansprechen kann. Dies hat den Vorteil, dass eine Semantik während des Betriebs eines Dialogsystems geändert werden kann, beispielsweise wenn ein Verkaufs- oder Bestelldialogsystem bei sich häufig ändernden Angeboten regelmäßig möglichst unterbre-

10 chungsfrei aktualisiert werden muss.

Bei einer vorteilhaften Ausführungsform dieses Verfahrens wird zur Spezifikation der Grammatik und der ihr zugeordneten Semantik eine objektorientierte Programmiersprache verwendet. Die formale Grammatik der zu erkennenden Sprachäußerungen kann dann als

15 Instanzen von Grammatik-Klassen spezifiziert werden, die generische Standard-Ersetzungsregeln realisieren und ihre gemeinsamen Eigenschaften und Funktionalitäten von einer oder mehreren grammatikalischen Basisklassen erben. Die Basisklassen stellen beispielsweise generische Parser-Methoden zur Verfügung, die bei der Spezifizierung der Grammatik an die konkret mit Terminal- und Nichtterminalwörtern instanziierten Ersetzungsregeln auf der Ebene

20 der Grammatik-Klassen angepasst werden müssen. Es ist sinnvoll, zur effizienten Spezifikation von Grammatiken Grammatik-Klassenhierarchien und/oder Grammatik-Klassenbibliotheken vorzusehen, die eine Vielzahl von möglichen Grammatiken bereits definieren und auf die im Bedarfsfalle zurückgegriffen werden kann.

25 Ebenso können die Basisklassen virtuelle Methoden zur Verfügung stellen, die bei der Verwendung einer attributierten Grammatik mit Methoden überschrieben werden können, die ein jeweils entsprechendes semantisches Objekt erzeugen. In diesem Falle wird beim Betrieb des Dialogsystems die semantische Umsetzung nicht zeitlich getrennt von der syntaktischen

Überprüfung durch das Applikationsprogramm durchgeführt, sondern die Semantik wird direkt während der Syntaxanalyse ausgeführt.

Bei einem erfindungsgemäßen Verfahren zum Erzeugen eines Dialogsystems mit einer Sprach-
schnittstelle, die gemäß der oben beschriebenen Verfahren entwickelt wurde, ist es vorteilhaft,
sowohl die Spracheingabeschnittstelle als auch das Applikationsprogramm in der gleichen -
möglicherweise objektorientierten – Übersetzersprache oder in einer auf die gleiche objekt-
orientierte Plattform abbildbaren Übersetzersprache zu schreiben. Daraus ergibt sich zwangs-
läufig, dass sowohl die formale Grammatik als auch die entsprechenden Programmmodule zur
Abbildung der Syntax einer Sprachäußerung auf die passende Semantik in dieser Sprache
implementiert werden.

Zur Erstellung einer solchen Spracheingabeschnittstelle gemäß dem genannten Verfahren wird
einem Entwickler oder Dienstleister vorzugsweise ein System zur Verfügung gestellt, dass zur
Spezifikation einer formalen Grammatik und einer passenden Semantik Syntaxspezifikations-
und Semantikdefinitions-Werkzeuge enthält. Mit dem Syntaxspezifikations-Werkzeug kann
mittels der oben beschriebenen Verfahren eine formale Grammatik spezifiziert werden, mittels
der gültige Sprachsignale identifiziert werden können. Das Semantikdefinition-Werkzeug
unterstützt einen Entwickler bei der Bereitstellung bzw. Programmierung der semantischen
Programmmodule und ihrer eindeutigen Zuordnung zu einzelnen Terminal- oder Nichtterminal-
wörtern der Grammatik. Die in Maschinensprache übersetzten Programmmodule sind dabei
unmittelbar von dem Applikationsprogramm ausführbar. Im Falle der Erzeugung von unmittel-
bar von der Applikation verwendbaren Datenstrukturen, werden diese von den in Maschinen-
sprache vorliegenden Teilprogrammen der Spracheingabeschnittstelle erzeugt

25

In einer besonders vorteilhaften Ausführungsform steht dem Grammatikentwickler dabei eine
graphische Entwicklungsoberfläche als Front-End des Syntaxspezifikations- und/der des
Semantikdefinitions-Werkzeugs zu Verfügung, die einen Grammatik-Editor und gegebenenfalls
einen Semantik-Editor besitzt. Sofern die Spracherkennung der Spracheingabeschnittstelle in

einer objektorientierten Übersetzersprache geschrieben ist, stellt der Grammatik-Editor einen erweiterten Klassen-Browser zur Verfügung, der die einfache Selektion von Basisklassen und Vererbung ihrer Funktionalitäten mit graphischen Mitteln (z.B. durch "drag-and-drop") ermöglicht. Die Instanziierung der Standard-Ersetzungsregeln durch Terminal- und Nichtterminal-
5 wörter und/oder Parsing-Methoden sowie gegebenenfalls durch Methoden zur Definition von semantischen Objekten kann dann über eine spezielle graphische Schnittstelle vorgenommen werden, die eine direkte Assoziation derartiger Daten mit der entsprechenden Grammatik-Klasse vornimmt und sie automatisch programmtechnisch umsetzt, also den entsprechenden Quellcode erzeugt. Zu besserer Unterscheidbarkeit von Basis-Klassen, abgeleiteten Klassen,
10 deren Methoden und semantischen Umsetzungen werden adäquate graphische Symbole eingesetzt.

Zur Programmierung der mitunter komplexen semantischen Programmmodule wird vorzugsweise eine Entwicklungsumgebung bereitgestellt, die beispielsweise aus Klassen-Browser,
15 Editor, Compiler, Debugger und einer Testumgebung besteht, die eine integrierte Entwicklung ermöglicht und die entsprechende Programmfragmente unter Umständen in die Grammatik-Klassen einkompiliert oder selbständige dynamische oder statische Bibliotheken erzeugt.

Die Erfindung wird im Folgenden unter Hinweis auf die beigefügten Zeichnungen anhand von
20 Ausführungsbeispielen näher erläutert. Es zeigen:

- Figur 1 einen Dialog eines Dialogsystems,
- Figur 2 eine Spezifikation einer formalen Grammatik,
- Figur 3 eine schematische Darstellung des Aufbaus eines Ausführungsbeispiels eines erfindungsgemäßen Dialogsystems mit einer Spracheingabeschnittstelle,
- 25 Figur 4a eine Definitionen von Grammatik-Klassen,
- Figur 4b eine Definition von Grammatik-Objekten als Instanzen von Grammatik-Klassen,
- Figur 5 eine semantische Umsetzung eines Grammatik-Objektes,
- Figur 6 einen graphischen Entwurf einer Grammatik.

Formal kann ein Dialogsystem als endlicher Automat beschrieben werden. Sein deterministisches Verhalten lässt sich mittels eines Zustands/Transitions-Diagramms beschreiben, das alle Zustände des Systems und die Ereignisse, die zu einem Zustandswechsel führen, die Transitionen, vollständig beschreibt. Figur 1 zeigt beispielhaft das Zustands/Transitions-Diagramm eines einfachen Dialogsystems 1. Dieses System kann zwei verschiedene Zustände S1 und S2 annehmen und besitzt vier Transitionen T1, T2, T3 und T4, die jeweils durch einen Dialogschritt D1, D2, D3 und D4 initiiert werden, wobei die Transition T1 den Zustand S1 auf sich selbst abbildet, während T2, T3 und T4 Zustandswechsel vornehmen. Der Zustand S1 ist der initiale bzw. Startzustand des Dialogsystems, den es am Ende eines jeden Dialogs mit dem Benutzer wieder einnimmt. In diesem Zustand generiert das System eine Startäußerung, die den Benutzer beispielsweise zu einer Äußerung auffordert: „Was kann ich für Sie tun?“. Dem Benutzer stehen nun die beiden Sprachäußerungen „Wie spät ist es?“ (Dialogschritt D1) und „Wie lautet die Wettervorhersage?“ (Dialogschritt D2) offen. Im Dialogschritt D1 antwortet das System mit der korrekten Uhrzeit und vollzieht anschließend die entsprechende Transition T1, indem es wieder in den Startzustand S1 zurückkehrt und erneut die Startäußerung ausgibt. Im Dialogschritt D2 antwortet das System zur genaueren Spezifikation des Benutzerwunsches mit der Gegenfrage „Für morgen oder für nächste Woche?“ und wechselt über die Transition T2 in den neuen Zustand S2. In Zustand S2 kann der Benutzer nur die Gegenfrage des Systems mit D3 „Für morgen!“ oder D4 „Für nächste Woche!“ beantworten, er hat hier nicht die Möglichkeit nach der Uhrzeit zu fragen. Auf die Klarstellung des Benutzers in den Dialogschritten D3 und D4 antwortet das System mit der jeweiligen Wettervorhersage und wechselt über die entsprechenden Transitionen T3 und T4 wieder in den Startzustand S1.

Um die einzelnen Dialogschritte durchführen zu können und adäquat auf die Äußerung des Benutzers reagieren zu können, ist es notwendig, die Sprachäußerung des Benutzers einerseits korrekt zu detektieren und anschließend in die von Benutzer gewünschte Reaktion umzusetzen, die Äußerung also zu verstehen. Natürlich ist es aus Gründen der Benutzerfreundlichkeit und Akzeptanz wünschenswert, dass das Dialogsystem in einem bestimmten Zustand

mehrere äquivalente Äußerungen eines Benutzers verarbeiten kann. Beispielsweise sollte das in Figur 1 beschriebene Dialogsystem bei der Transition T1 nicht nur den einen spezifischen Dialogschritte D1 verstehen, sondern auch auf synonyme Anfragen wie z.B. „Wie viel Uhr ist es?“ oder „Welche Uhrzeit haben wir?“ richtig reagieren. Darüber hinaus stellen realistische Systeme in einem Zustand oft eine große Anzahl von möglichen Dialogschritten zur Verfügung, die eine Vielzahl von unterschiedlichen Transitionen initiieren. Abgesehen von der trivialen und meist impraktikablen Lösung, im System alle möglichen Dialogschritte zum Vergleich mit der jeweiligen Benutzeranfrage zusammen mit den entsprechenden Systemreaktionen zu hinterlegen, ist es in derartigen Fällen sinnvoll, die möglichen Äußerungen eines Benutzers durch eine formale Grammatik GR zu spezifizieren.

Figur 2 zeigt ein Beispiel einer formalen Grammatik GR zur sprachlichen Steuerung eines Gerätes. Die Grammatik GR besteht aus den Nichtterminalwörtern <command>, <play>, <stop>, <goto> und <lineno>, den Terminalwörtern „play“, „go“, „start“, „stop“, „halt“, „quit“, „go to line“, „1“, „2“ und „3“ und Ersetzungsregeln AR und KR, die für jedes Nichtterminalwort eine Ersetzung durch Nichtterminal- und/oder Terminalwörtern vorschreibt. Entsprechend ihrer Funktionsweise werden die Ersetzungsregeln in Alternativregeln AR und Kettenregeln KR unterschieden, wobei das Startsymbol <command> von einer Alternativregel abgeleitet wird. Eine Alternativregel AR ersetzt ein Nichtterminalwort durch eine der genannten Alternativen, und eine Kettenregel KR ersetzt ein Nichtterminalwort durch eine Reihung von weiteren Terminal- oder Nichtterminalwörtern. Beginnend mit der initialen Ersetzung des Startwortes <command> können alle gültigen Sätze – also gültige Reihungen von Terminalwörtern – der durch die formale Grammatik GR spezifizierten Sprache in Form eines Ableitungs- bzw. Ersetzungsbaums generiert werden. So wird durch die sequentielle Ersetzung der Nichtterminalsymbole <command>, <goto>, und <lineno> beispielsweise der Satz „go to line 2“ erzeugt und als gültige Sprachäußerung definiert, nicht aber den Satz „proceed to line 4“. Diese Ableitung eines konkreten Satzes aus dem Startwort repräsentiert den Schritt der Syntaxanalyse.

Da die in Figur 2 genannte Grammatik GR eine attributierte Grammatik ist, erlaubt sie die direkte Abbildung der Syntax auf die Semantik, d.h. auf von der Applikation 3 ausführbaren/interpretierbaren Kommandos. Diese sind bereits in der Grammatik GR für jedes einzelne Terminalwort – angegeben in den geschweiften Klammern – spezifiziert. Die in der

5 Syntaxanalyse SA als gültig erkannte Äußerung „go to line 2“ wird in die Kommandos „GOTO TWO“ semantisch umgesetzt. Durch die Abbildung mehrerer syntaktische Konstrukte auf die gleiche Semantik werden auch synonyme Äußerungen berücksichtigt. So werden beispielsweise die Äußerungen „play“, „go“ und „start“ auf das gleiche Kommando „PLAY“ semantisch abgebildet und führen zur gleichen Reaktion des Dialogsystems 1.

10

Ein Ausführungsbeispiel eines Dialogsystems 1 mit einer erfindungsgemäßen Spracheingabeschnittstelle 2 und einer mit der Spracheingabeschnittstelle zusammenwirkenden Applikation 3 ist in Figur 3 dargestellt. Die Applikation 3 umfasst eine Dialogsteuerung 8, die das Dialogsystem 1 gemäß der in einem Zustands/Transitions-Diagramm festgelegten Zustände, Transi-

15 tionen und Dialoge steuert.

15

Eine eingehende Sprachäußerung wird nun zunächst wie üblich von einer Signalaufnahme-Einheit 4 der Spracheingabeschnittstelle 2 in ein digitales Audiosprachsignal AS umgesetzt. Der eigentliche Prozess der Spracherkennung wird dabei von der Dialogsteuerung 8 durch ein

20 Startsignal ST initiiert.

20

Der in der Spracheingabeschnittstelle 2 integrierte Spracherkenner 5 umfasst eine Syntaxanalyse-Einheit zur Durchführung der Syntaxanalyse SA und eine Semantiksynthese-Einheit zur Durchführung der anschließenden Semantiksynthese SS. Die im Syntaxanalyse-Schritt zu

25 überprüfende formale Grammatik GR (oder eine von ihr abgeleitete Datenstruktur, die von der Syntaxanalyse direkt verwendet wird) wird gemäß dem aktuellen Zustand des Dialogsystems 1 und der zu erwartenden Dialoge von der Dialogsteuerung 8 jeweils an die Syntaxanalyse-Einheit 6 übergeben. Gemäß dieser Grammatik GR wird das Audiosprachsignal AS verifiziert und im Falle der Gültigkeit von der Semantiksynthese-Einheit 7 auf seine Semantik abgebildet.

Es existieren zwei Varianten zur Definition von Semantik. Soweit nicht anders abweichend erwähnt, wird im folgenden davon ausgegangen, dass es sich ohne Beschränkung der Erfindung bei dem Erkennungsergebnis ER um ein oder mehrere Programmmodule handelt.

- 5 Hierbei ergibt sich die Semantik direkt aus der direkten Zuordnung von Terminal- und Nichtterminalsymbolen zu maschinensprachlichen Programmmodulen PM, die von einer Programmausführungseinheit 9 der Applikation 3 ausgeführt werden können. Die maschinensprachlichen Programmmodule PM aller Terminal- und Nichtterminalwörter einer vollständig abgeleiteten Sprachäußerung werden dabei von der Semantiksynthese-Einheit 7 zu einem
- 10 maschinensprachlichen Erkennungsergebnis ER zusammengeführt und der Programmausführungseinheit 9 der Applikation 3 zur Ausführung bereitgestellt bzw. ihr als unmittelbar ausführbares Maschinenprogramm übergeben.

- Zur vollständigen Beschreibung der Erfindung sei zusätzlich erläutert, dass in einer zweiten
- 15 Variante den Terminal- und Nichtterminalwörtern auch Datenstrukturen zugeordnet werden können, die unmittelbar von maschinensprachlichen Programmteilen der Spracheingabeschnittstelle 2 generiert werden und die ein Erkennungsergebnis ER repräsentieren. Diese Datenstrukturen können dann ohne weitere interne Umsetzung, Transformation oder Interpretation von der Applikation 3 verwendet werden. Es ist darüber hinaus auch möglich, die
- 20 beiden genannten Varianten zu kombinieren, so dass die Semantik zum Teil durch maschinensprachliche Programmmodule und zum Teil durch unmittelbar von der Applikation verwendbare Datenstrukturen definiert wird.

- Sowohl die Spracherkennung 5 der Spracheingabeschnittstelle 2 als auch das Applikations-
- 25 programm 3 sind hier in der gleichen objektorientierten Übersetzersprache geschrieben, oder ein Sprachen, die auf der gleichen objektorientierten Plattform ablauffähig sind. Die Übergabe des Erkennungsergebnisses ER ist somit sehr einfach durch Übermittlung von Referenzen bzw. Zeigern möglich. Die Verwendung einer objektorientierten Übersetzersprache ist – insbesondere auch bei der oben genannten Kombination von semantischen Programmmodulen und

Datenstrukturen – besonders vorteilhaft. Durch den objektorientierten Programmentwurf werden zudem sowohl die Grammatiken GR als auch das Erkennungsergebnis ER in Form von programmiersprachlichen Objekten als Instanzen von Grammatik-Klassen GK bzw. als Methoden dieser Klassen realisiert. Die Figuren 4a, 4b und 5 zeigen diesen Prozess im Detail.

5

Ausgehend von der Definition der formalen Grammatik GR in Figur 2 zeigt Figur 4a die Implementierung geeigneter Grammatik-Klassen GK zur Umsetzung der formalen Definition in eine objektorientierte Programmiersprache. Sämtliche Grammatik-Klassen GK werden hier von einer abstrakten grammatikalischen Basisklasse BK abgeleitet, die ihre Methode an die von ihr abgeleiteten Grammatik-Klassen GK vererbt. In dem in Figur 4a dargestellten Ausführungsbeispiel handelt es sich um drei verschiedene abgeleitete Grammatik-Klassen GK, die als mögliche prototypische Ersetzungsregeln in Form einer Terminalregel TR, eine Alternativregel AR und eine Kettenregel KR implementiert werden.

- 15 Die abstrakte Basisklasse BK verlangt die Methoden GetPhaseGrid(), Value() und PartialParse(), wobei die Methode GetPhaseGrid() zur Initialisierung des signaltechnischen Spracherkennungsprozesses verwendet wird und zum Verständnis des syntaktischen Erkennungsprozesses nicht betrachtet werden muss. Abgesehen von GetPhaseGrid() ist die einzige von außen anzusprechende Funktionalität die Methode Value(), die den ihr mit dem Argument
- 20 „phrase“ übergebenen Satz evaluiert und somit den Zugriff auf die zentrale Parsing-Funktionalität realisiert. Value() gibt als Ergebnis die Semantik zurück. In einem einfachen Fall kann dies eine Liste sein, die die erkannten syntaktischen Einheiten des Satzes separat aufführt. Gemäß der formalen Grammatik GR aus Figur 2 würde beispielsweise für die Phrase „go to line 2“ die Liste („go to line“, „2“) erzeugt werden. In anderen Fällen können die Daten weiter
- 25 verarbeitet werden, wie in dem oben aufgeführten Beispiel der Zeit-Grammatik. Der Mechanismus dazu wird weiter unten beschrieben. Dieses Ergebnis der Syntaxanalyse SA wird anschließend semantisch in ein Maschinenspracheprogramm bzw. in eine Datenstruktur umgesetzt und der Applikation 3 zur unmittelbaren Ausführung/Verwendung bereitgestellt. Da die Arbeitsweise der übergeordneten Parsing-Methode Value() von den jeweils anwendbaren

Ersetzungsregeln abhängt, greift Value() intern auf die abstrakte Methode PartialParse() zurück. Diese kann jedoch nicht bereits in der Basisklasse BK realisiert werden, sondern wird erst durch die jeweilige abgeleitete Grammatik-Klasse GK implementiert.

- 5 Die in der Basisklasse BK verlangte Parsing-Funktionalität der PartialParse()-Methode wird also erst in den Grammatikklassen GK implementiert. Neben dieser regelabhängigen Parsing-Methode besitzen die abgeleiteten Grammatikklassen GK spezifische sogenannte Konstruk-
- toren (PhraseGrammar(), ChoiceGrammar(), ConcatenatedGrammar()), mit denen zur Laufzeit der Syntaxanalyse SA Instanzen dieser Klassen, also Grammatik-Objekte GO
- 10 erzeugt werden können. Die abgeleiteten Grammatik-Klassen TR, AR, und KR stellen also das programmiersprachliche „Gerüst“ für die Realisierung einer konkreten Ersetzungsregel einer bestimmten formalen Grammatik GR dar. Der Konstruktor der Terminalregel TR
- PhraseGrammar benötigt lediglich das Terminalwort, durch das ein bestimmtes Nichtterminal-
- wort zu ersetzen ist. Der Konstruktor der Alternativregel AR ChoiceGrammar benötigt eine
- 15 Liste mit den möglichen alternativen Ersetzungen, während der Konstruktor der Kettenregel
- KR ConcatenatedGrammar eine Liste von aneinanderzureihenden Terminal- und/oder
- Nichtterminalwörtern benötigt. Jede dieser drei Grammatik-Klassen GK realisiert auf
- individuelle Art und Weise die abstrakte PartialParse()-Methode der Basisklasse BK.
- 20 Ausgehend von den in Figur 4a definierten Grammatik-Klassen GK zeigt Figur 4b als Beispiel die Verwendung dieser Klassen zur Realisierung der in Figur 2 angegebenen Grammatik GR
- durch die Erzeugung (Instanziierung) von Grammatik-Objekten GO. Das Command-Objekt
- wird durch Instanziierung derjenigen Grammatik-Klasse GK zur Laufzeit erzeugt, die die
- Alternativregel AR realisiert. Seine Funktion ist, das nichtterminale Startwort <command>
- 25 durch eines der Nichtterminalwörter <play>, <stop> oder <goto> zu ersetzen, die dem
- Konstruktor der jeweiligen Alternativregel AR als Argumente mitgegeben werden.

Das Play-Objekt wird ebenfalls durch das Aufrufen des Konstruktors der Alternativregel AR erzeugt. Anders als beim Konstruktoraufruf des Command-Objekts enthält das Argument der

- Konstruktoraufrufs des Play-Objekts keine Nichtterminalwörter, sondern ausschließlich Terminalwörter. Die Terminalwörter werden durch einen geschachtelten Aufruf des Konstruktors der Terminalregel TR angegeben und realisieren die Wörter „play“, „go“, und „start“. Ebenso werden die Ersetzungsregeln der Nichtterminalwörter <stop> und <lineno> durch
- 5 entsprechende Aufrufe des Konstruktors der Alternativregel AR erzeugt. Das Goto-Objekt wird schließlich als Instanz derjenigen Grammatik-Klasse GK erzeugt, die die Kettenregel KR implementiert. Der Konstruktor enthält das Terminalwort „go to line“ und das Nichtterminalwort <lineno> als Argument.
- 10 Zur semantischen Umsetzung einer durch die Grammatik-Objekte GO evaluierten Äußerung werden bei der formalen Grammatik GR aus Figur 2 lediglich die Terminalwörter in Programmmodule PM umgesetzt, der Applikation 3 als Referenzen übergeben und unmittelbar von ihr ausgeführt. Die Programmmodule PM oder entsprechende Referenzierungen sind mit den Terminalwörtern direkt über die Definition der Grammatik GR assoziiert (siehe Fig. 2). In
- 15 der konkreten Ausführungssituation sieht dies beispielsweise wie folgt aus: jede der <command>-Regeln erzeugt ein Command-Objekt, dessen Execute()-Methode von der Applikation 3 direkt ausgeführt werden kann. Die goto-Regel würde dabei ein spezielles Command-Objekt erzeugen, dass auch die entsprechende Zeilennummer enthält.
- 20 Im Gegensatz zu dieser strikten Trennung zwischen Syntaxanalyse SA, Semantiksyntaxese SS und Ausführen des semantischen Maschinenspracheprogramms zeigt Figur 5 eine direkte Synthese der semantischen Anweisungen und ihre Ausführung durch die Spracheingabeschnittstelle 2 am Beispiel eines Grammatik-Objektes GO, das eine Multiplikation durch eine Kettenregel KR realisiert. Das Multiplikation-Objekt wird instanziiert als Aneinanderreihung
- 25 von drei Elementen: einer natürlichen Zahl zwischen 1 und 9 (die Klasse NumberGrammar kann beispielsweise durch Vererbung aus der Klasse ChoiceGrammar entstehen), dem Terminalwort „times“ und einer erneuten natürlichen Zahl aus dem Intervall 1 bis 9. Anstatt als Ergebnis des Parsens die Liste („3“, „times“, „5“) zur semantischen Umsetzung zu übergeben,

kann die Anweisung „3 times 5“ direkt im Objekt ausgeführt und das Ergebnis 15 zurückgegeben werden. Die Berechnung wird dabei im vorliegenden Beispiel durch einen speziellen Synthese-Eventhandler SE vorgenommen, der die Daten des Multiplikation-Objektes – im vorliegenden Beispiel also die beiden Faktoren der Multiplikation – entnimmt und verknüpft.

5

Eine derartige effiziente und mit der Syntaxanalyse SA verzahnte Semantik-Synthese SS wird allein durch die erfindungsgemäße Implementierung der Semantik eines syntaktischen Konstrukts in einer Übersetzersprache und Übersetzung in unmittelbar ausführbare maschinensprachliche Programmmodule PM ermöglicht, da nur so die Semantiksyntaxese SS direkt in die Syntaxanalyse SA integriert werden kann. Durch die Verwendung einer objektorientierten anstelle einer prozeduralen/imperativen Programmiersprache können die verwendeten Datenstrukturen zusätzlich für Dienstleister und Endanwender geeignet strukturiert und gekapselt werden, während der Datentransfer zwischen Syntaxanalyse und Semantiksyntaxese effizient kontrollierbar ist.

10

15

Eine spezielle Funktionalität von Designwerkzeugen zum Grammatikentwurf wird anhand von Figur 6 am Beispiel einer Zeitgrammatik erläutert. Zum Entwurf einer speziellen Grammatik werden die von den Grammatik-Klassen GK vorgegebenen Ersetzungsregeln KR, AR und TR graphisch kombiniert und durch Einsetzen entsprechender Terminal- und Nichtterminalwörter instanziiert, also entsprechende Grammatik-Objekte GO erzeugt.

20

Die verschiedenen Ersetzungsregeln werden daher in Figur 6 durch unterschiedliche Formen der Boxen des Flussdiagramms unterschieden. Nach der graphischen Selektion einer bestimmten Ersetzungsregel wird zur Spezifikation (also zur Instanziierung der Regel) beispielsweise durch Doppelklick oder eine beliebige andere Benutzeraktion ein Grammatik-editor geöffnet, in welchem zur Spezifikation der Sub-Grammatik, je nach ausgewählter Regel, die Alternativen, die Reihungen oder die Terminalwörter angegeben werden können. Nach der Spezifikation der entsprechenden Sub-Grammatik wird der Unterbaum wieder geschlossen und die spezifizierte Teil-Grammatik erscheint in formaler Schreibweise in der

25

übergeordneten Box. Um komplexe Grammatiken zu ermöglichen, können zur Spezifizierung einer Sub-Grammatik weitere Regeln in sie eingesetzt werden.

- Im Beispiel der Uhrzeit-Grammatik beginnt der Entwurf mit der Auswahl einer Alternativregel
- 5 AR, die mit vier Sub-Grammatiken in Form von alternativen Kettenregeln KR – angezeigt durch die ovalen Boxen – belegt wird.

- Für die erste und vierte der Alternativen sind die Bäume der Sub-Grammatiken geschlossen, durch einen Doppelklick auf die entsprechenden Boxen bzw. durch eine entsprechende
- 10 Aktion können sie jedoch sichtbar gemacht werden. So würde bei der vierten Alternative ((1..20|quarter)(minutes to|to)1..12) durch einen Doppelklick o.Ä. auf die Kettenregel-Box KR eine Reihung aus zwei Alternativregeln AR und einer Terminalregel TR sichtbar.

- Für die zweite und dritte der Alternativen sind die Bäume der Sub-Grammatiken teilweise
- 15 sichtbar. Die zweite Alternative ((1..12(1..59|)))(AM|PM|)) besteht aus einer Reihung aus der Kettenregel (1..12(1..59|)) und der Alternativregel (AM|PM|). Die Kettenregel KR besteht wiederum aus einer Reihung aus einer Terminalregel TR und einer Alternativregel AR, die zwei alternative Terminalregeln TR beinhaltet. Die Alternativregel AR bietet drei verschiedene
- 20 Terminalregeln TR als Alternativen an, die die Terminalwörter „AM“, „PM“ und ein noch nicht spezifiziertes drittes Terminalwort einsetzen. Durch Doppelklick o.Ä. auf eine Terminalregel TR können die abschließend einzusetzenden Terminalwörter, also das Vokabular der formalen Sprache, angegeben werden. Auf diese Art und Weise kann mit dem Grammatikeditor eine beliebige Grammatik GR in der gewünschten Komplexität spezifiziert und graphisch dargestellt werden.

25

Die auf diese Weise graphisch spezifizierte formale Grammatik wird nun vollständig und automatisch in entsprechende programmiersprachliche Grammatik-Klassen GK einer objektorientierten Übersetzersprache umgesetzt, die nach der Übersetzung zur Laufzeit des

Dialogsystems 1 instanziiert werden und als Ersetzungsregeln die Gültigkeit einer Sprachäußerung durch Ableiten/Parsen verifizieren.

- 5 Durch Aktivierung einer entsprechenden Funktion eines Semantikers kann automatisch ein Eventhandler SE zur Semantik- bzw. Attributsynthese erzeugt werden. Es öffnet sich dann automatisch ein Editorfenster, in dem der entsprechende Programmcode für den Event in der objektorientierten Übersetzersprache ergänzt werden kann. Nach ihrer Übersetzung kann die so spezifizierte Grammatik-Klasse der Applikation in Form einer statisch oder dynamisch zu
- 10 bindenden Bibliothek zur Ausführung bereitgestellt werden.

- Es wird abschließend noch einmal darauf hingewiesen, dass es sich bei der in den Figuren und der Beschreibung dargestellten Spracheingabeschnittstelle und dem Dialogsystem lediglich um Ausführungsbeispiele handelt, die vom Fachmann in einem weiten Umfang variiert werden
- 15 können, ohne den Rahmen der Erfindung zu verlassen. Insbesondere können die Programmfragmente, die bei den gezeigten Ausführungsbeispielen in der objektorientierten Programmiersprache C# verfasst sind, in jeder beliebigen anderen objektorientierten Programmiersprache oder sogar in anderen imperativen Programmiersprachen geschrieben werden. Es wird außerdem der Vollständigkeit halber darauf hingewiesen, dass die Verwendung der unbestimmten
- 20 Artikel „ein“ bzw. „eine“ nicht ausschließt, dass die betreffenden Merkmale auch mehrfach vorhanden sein können, und dass die Verwendung des Begriffs „umfassen“ nicht die Existenz weiterer Elemente oder Schritte ausschließt.

PATENTANSPRÜCHE

1. Verfahren zum Betrieb eines Dialogsystems (1) mit einer Spracheingabeschnittstelle (2) und einer mit der Spracheingabeschnittstelle (2) zusammenwirkenden Applikation (3), bei dem die Spracheingabeschnittstelle (3) Audiosprachsignale (AS) eines Benutzers erfasst und direkt in ein Erkennungsergebnis (ER) in Form von Binärdaten umsetzt, die unmittelbar von der
- 5 Applikation (3) verwendbar sind.
2. Verfahren nach Anspruch 1,
dadurch gekennzeichnet,
dass die Binärdaten zumindest ein in Maschinensprache vorliegendes, unmittelbar von der
- 10 Applikation (3) ausführbares Programmmodul (PM) in Form eines Objekts einer objektorientierten Übersetzersprache und/oder ein Datenobjekt einer objektorientierten Übersetzersprache umfassen.
3. Verfahren nach Anspruch 1 oder 2,
- 15 dadurch gekennzeichnet,
dass beim Umsetzen des Audiosprachsignals (AS) in ein Erkennungsergebnis (ER) zunächst in einem Syntaxanalyse-Schritt (SA) die dem Audiosprachsignals (AS) entsprechende Phrase aufgrund einer formalen Grammatik (GR) detektiert wird, wobei das gültige Vokabular des Audiosprachsignals (AS) den Terminalwörtern der formalen Grammatik (GR) entspricht, und
- 20 anschließend das Erkennungsergebnis (ER) in einem Semantiksynthese-Schritt (SS) aus den den Terminalwörtern zugeordneten, in Maschinensprache vorliegenden ausführbaren Programmmodulen (PM) generiert wird.

4. Verfahren nach Anspruch 3,
dadurch gekennzeichnet,
dass die Grammatik (GR) vor dem Beginn eines Dialogs vollständig definiert wird und
während des Dialogs unveränderbar ist.
- 5
5. Verfahren nach Anspruch 3,
dadurch gekennzeichnet,
dass die Grammatik (GR) während eines Dialogs dynamisch geändert wird.
- 10 6. Verfahren nach einem der Ansprüche 3 bis 5,
dadurch gekennzeichnet,
dass die Grammatik (GR) Ersetzungsregeln (AR, KR) umfasst, die als objektorientierte
Grammatik-Klassen (GK) realisiert sind, die jeweils eine regelabhängige Parsing-Funktion als
Methode besitzen.
- 15
7. Verfahren nach einem der Ansprüche 3 bis 6,
dadurch gekennzeichnet,
dass die Grammatik (GR) in Form von zumindest einem Grammatik-Objekt (GO) als Instanz
zumindest einer objektorientierten Grammatik-Klasse (GK) spezifiziert ist und beim
- 20 Syntaxanalyse-Schritt (SA) das Audiosprachsignal (AS) gemäß von Ersetzungsregeln (AR,
KR) der Grammatik (GR) überprüft wird.
8. Verfahren nach einem der Ansprüche 3 bis 7,
dadurch gekennzeichnet,
- 25 dass der Syntaxanalyse-Schritt (SA), der Semantiksynthese-Schritt (SS) und/oder eine
Verwendung/Ausführung des Erkennungsergebnisses (ER) zumindest teilweise zeitlich
überlappend erfolgen.

9. Verfahren nach einem der Ansprüche 6 bis 8,

dadurch gekennzeichnet,

dass ein das Erkennungsergebnis erzeugender Programmteil der Spracheingabeschnittstelle als

- 5 Methode einer objektorientierten Klasse, insbesondere als Methode des Grammatik-Objektes (GO) eingebunden wird.

10. Verfahren nach einem der Ansprüche 6 bis 8,

dadurch gekennzeichnet,

- 10 dass das Erkennungsergebnis (ER) von einer Methode einer Grammatik-Klasse (GK) definiert und von dieser als Objekt zurückgegeben wird.

11. Verfahren zur Erstellung einer Spracheingabeschnittstelle (2) für ein Dialogsystem (1) mit einer mit der Spracheingabeschnittstelle (2) zusammenwirkenden Applikation (3), umfassend

15 die Schritte:

- Spezifizieren gültiger Spracheingabesignale (AS) durch eine formale Grammatik (GR), wobei das gültige Vokabular der Spracheingabesignale in Form von Terminalwörtern der Grammatik (GR) definiert wird,
- Bereitstellen von Binärdaten, die die Semantik gültiger Audiosprachsignale (AS)
- 20 repräsentieren und die aus zur Systemlaufzeit von der Applikation (3) unmittelbar verwendbaren und von einem Programmteil der Spracheingabeschnittstelle (2) erzeugten Datenstrukturen und/oder unmittelbar von der Applikation (3) ausführbaren Programmmodulen (PM) bestehen, und/oder Bereitstellen von Programmteilen, welche die Binärdaten erzeugen,
- 25 - Zuordnen der Binärdaten und/oder Programmteilen zu einzelnen oder Kombinationen von Terminalwörtern oder Nichtterminalwörtern zur Abbildung eines gültigen Audiosprachsignals (AS) auf eine passende Semantik,

- Übersetzen der Programmteile und/oder Programmmodule (PM) in Maschinensprache so, dass die übersetzten Programmteile beim Betrieb des Dialogsystems (1) unmittelbar von der Applikation (3) nutzbare Datenstrukturen erzeugen bzw. die übersetzten Programmmodule (PM) beim Betrieb des Dialogsystems (1) unmittelbar von der Applikation (3) ausführbar sind.

12. Verfahren nach Anspruch 11,
dadurch gekennzeichnet,
dass die formale Grammatik (GR) durch zumindest ein Grammatik-Objekt (GO) als Instanz
10 zumindest einer objektorientierten Grammatik-Klasse (GK) spezifiziert wird.

13. Verfahren nach Anspruch 12,
dadurch gekennzeichnet,
dass zumindest eine Grammatik-Klasse (GK) durch Vererbung aus einer oder mehreren
15 vorgegebenen Klassen einer Grammatik-Klassenhierarchie und/oder einer Grammatik-Klassenbibliothek abgeleitet wird.

14. Verfahren nach einem der Ansprüche 11 bis 13,
dadurch gekennzeichnet,
20 dass die Programmmodule (PM) in einer objektorientierten Übersetzersprache programmiert werden.

15. Verfahren nach einem der Ansprüche 12 bis 14,
dadurch gekennzeichnet,
25 dass zumindest eine Grammatik-Klasse (GK) und/oder die Programmmodule (PM) in Maschinensprache übersetzt werden und als statisch und/oder dynamisch zu bindende Bibliothek bereitgestellt werden.

16. Verfahren nach einem der Ansprüche 11 bis 15,

dadurch gekennzeichnet,

5 dass die Spezifikation der formalen Grammatik (GR) mit Hilfe eines graphischen Grammatik-Editors erfolgt und/oder die Definition der Semantik mit Hilfe eines graphischen Semantik-Editors erfolgt.

17. Verfahren nach Anspruch 16,

dadurch gekennzeichnet,

10 dass die Spezifikation der formalen Grammatik (GR) mit Hilfe des graphischen Grammatik-Editors durch Selektieren und/oder Ableiten von vorgegebenen Grammatik-Klassen (GK) und Belegen der Grammatik-Klassen mit Ersetzungsregeln (AR, KR) und/oder Terminalwörtern und/oder Nichtterminalwörtern erfolgt, wobei jeder Grammatik-Klasse (GK) und/oder jeder Ersetzungsregel (AR, KR) ein graphisches Symbol zugeordnet ist.

15

18. Verfahren nach Anspruch 16 oder 17,

dadurch gekennzeichnet,

20 dass der graphische Semantik-Editor zur Definition der Semantik der formalen Grammatik (GR) für jedes Programmmodul (PM) ein Editor-Fenster zum Erstellen des Programmmoduls (PM) bereitstellt und das Programmmodul mit einem Terminal- oder Nichtterminalwort assoziiert.

19. Verfahren zum Erzeugen eines Dialogsystems (1) mit einer Spracheingabeschnittstelle (2) und einer Applikation (3), wobei die Spracheingabeschnittstelle (2) mit einem Verfahren nach

25 einem der Ansprüche 10 bis 17 erzeugt wurde.

20. Verfahren nach Anspruch 19,

dadurch gekennzeichnet,

dass die Spracheingabeschnittstelle (2), die Applikation (3) und gegebenenfalls die zum

- 5 Erkennungsergebnis (ER) gehörenden Programmmodule (PM) jeweils zumindest teilweise in der gleichen objektorientierten Übersetzersprache geschrieben werden oder auf der gleichen objektorientierten Plattform ablauffähig sind.

21. Spracheingabeschnittstelle (2) für ein Dialogsystem (1) zur sprachlichen Steuerung eines

- 10 Gerätes oder Prozesses durch einen Benutzer, die mit einer Applikation (3) des Dialogsystems (1) zusammenwirkt und Audiosprachsignale (AS) erfasst und diese direkt in ein Erkennungsergebnis (ER) in Form von unmittelbar von der Applikation (3) verwendbaren Binärdaten umsetzt.

- 15 22. Dialogsystem (1), umfassend eine Spracheingabeschnittstelle (2) nach Anspruch 20.

23. System zur Erstellung einer Spracheingabeschnittstelle (2) eines Dialogsystems (1), umfassend

ein Syntaxspezifikations-Werkzeug, mit dem gültige Audiosprachsignale (AS) des

- 20 Dialogsystems (1) durch eine formale Grammatik (GR) spezifiziert werden, wobei das gültige Vokabular der Audiosprachsignale (AS) in Form von Terminalwörtern der Grammatik (GR) definiert wird, und

ein Semantikdefinitions-Werkzeug zum Bereitstellen von Programmmodulen (PM) und Zuordnen der Programmmodule (PM) zu einzelnen oder Kombinationen von

- 25 Terminalwörtern, derart dass nach einer Übersetzung in Maschinensprache die übersetzten Programmmodule (PM) beim Betrieb des Dialogsystems (1) von der Applikation (3) unmittelbar ausgeführt werden können.

24. System nach Anspruch 23,

gekennzeichnet durch

eine objektorientierte Grammatik-Klassenbibliothek und/oder eine objektorientierte

- 5 Grammatik-Klassenhierarchie, so dass die formale Grammatik (GR) als Instanz einer der Grammatik-Klassenbibliothek entnommenen oder von den Klassen der Grammatik-Klassenbibliothek abgeleiteten Grammatik-Klasse (GK) und/oder als Instanz einer der Grammatik-Klassenhierarchie entnommenen oder von den Klassen der Grammatik-Klassenhierarchie abgeleiteten Grammatik-Klasse (GK) spezifiziert ist.

10

25. System nach Anspruch 23 oder 24,

gekennzeichnet durch

einen graphischen Grammatik-Editor zur Spezifikation der formalen Grammatik (GR) und/oder einen graphischen Semantik-Editor zur Definition der Semantik.

15

ZUSAMMENFASSUNG

Spracheingabeschnittstelle für Dialogsysteme

- Es wird ein Verfahren zum Betrieb eines Dialogsystems (1) mit einer
- 5 Spracheingabeschnittstelle (2) und einer mit der Spracheingabeschnittstelle (2) zusammenwirkenden Applikation (3) beschrieben. Dabei erfasst die Spracheingabeschnittstelle (2) Audiosprachsignale (AS) eines Benutzers und setzt diese in ein
- Erkennungsergebnis (ER) in Form von unmittelbar von der Applikation verwendbaren Binärdaten. Dieses Erkennungsergebnis (ER) wird der Applikation (3) bereitgestellt.
- 10 Außerdem werden ein Verfahren und ein System zur Erstellung einer entsprechenden Spracheingabeschnittstelle (2) sowie eine Spracheingabeschnittstelle (2) und ein Dialogsystem (1) mit einer derartigen Spracheingabeschnittstelle (2) beschrieben.

Fig. 3

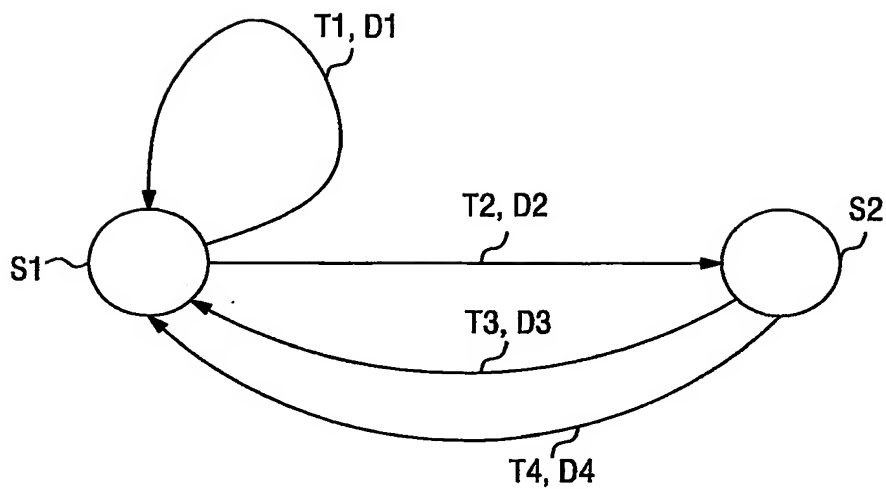


FIG. 1
(Stand der Technik)

AR \rightarrow <command> = <play> | <stop> | <goto>;
 AR \rightarrow <play> = (play|go|start) {PLAY};
 AR \rightarrow <stop> = (stop|halt|quit) {STOP};
 KR \rightarrow <goto> = go to line <lineno> {GOTO};
 AR \rightarrow <lineno> = 1 {ONE} | 2 {TWO} | 3 {THREE};

} \leftarrow GR

FIG. 2
(Stand der Technik)

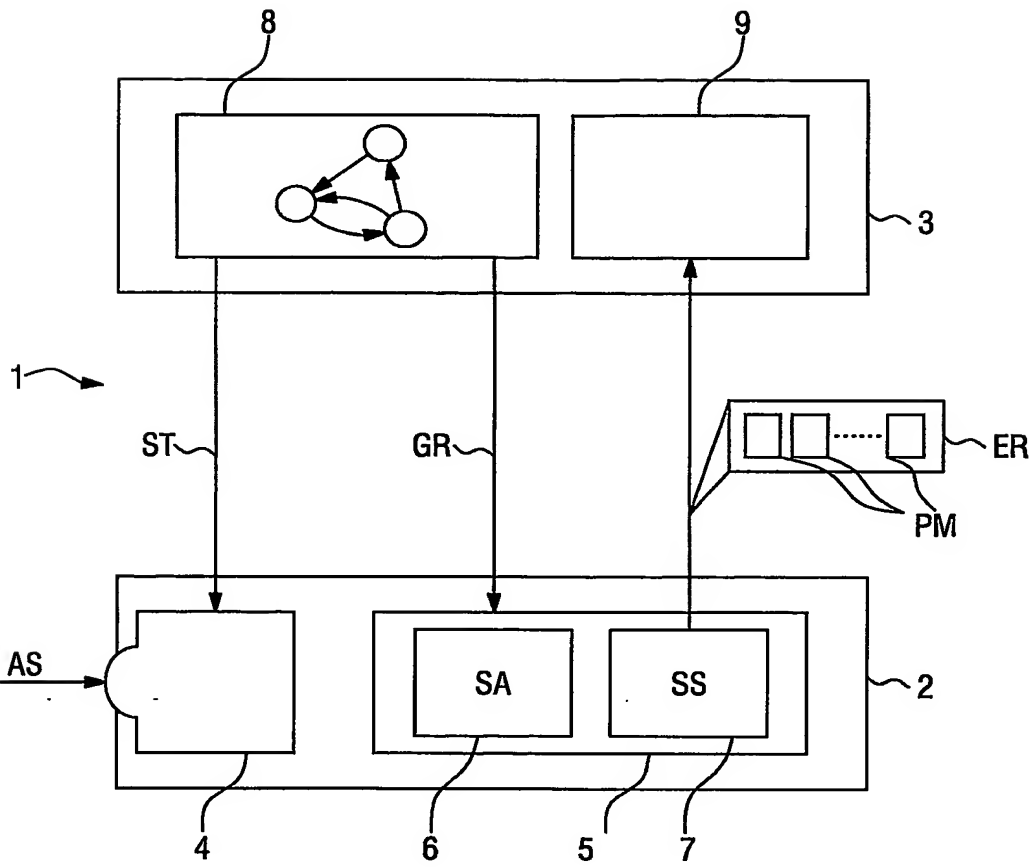


FIG. 3

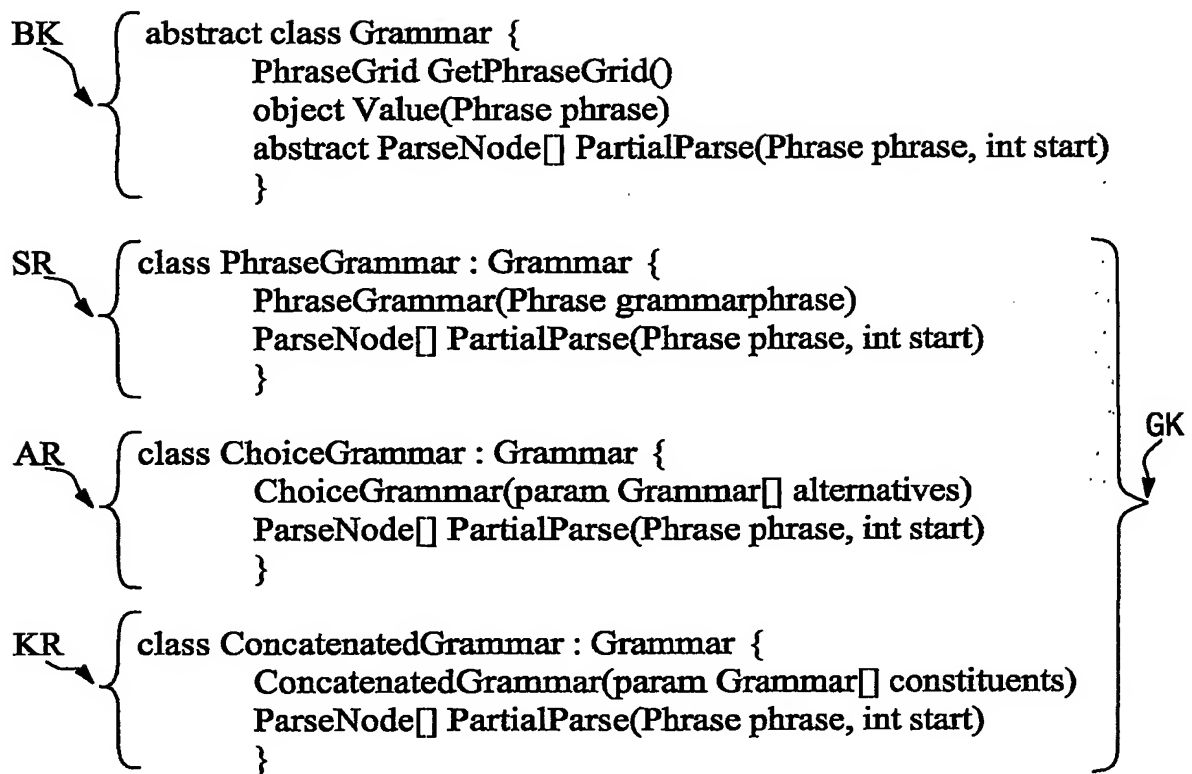


FIG. 4a


AR→ Grammar command = new ChoiceGrammar(play, stop, goto)

AR→ Grammar play = new ChoiceGrammar(
 new PhraseGrammar("play"),
 new PhraseGrammar("go"),
 new PhraseGrammar("start")
)

AR→ Grammar stop = new ChoiceGrammar(
 new PhraseGrammar("stop"),
 new PhraseGrammar("halt"),
 new PhraseGrammar("quit running")
)

AR→ Grammar lineno = new ChoiceGrammar(
 new PhraseGrammar("1"),
 new PhraseGrammar("2"),
 new PhraseGrammar("3")
)

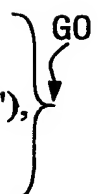
KR→ Grammar goto = new ConcatenatedGrammar(
 new PhraseGrammar("go to line"),
 lineno
)



GO

FIG. 4b

```
KR  $\leadsto$  Grammar multiplication = new ConcatenatedGrammar(  
    new NumberGrammar(1,9),  
    new PhraseGrammar("times"),  
    new NumberGrammar(1,9)  
)
```



SE \leadsto void OnSynthesize(SynthesizeEventArgs e) {
 e.Value = e.Values[0] * e.Values[2]
}

FIG. 5

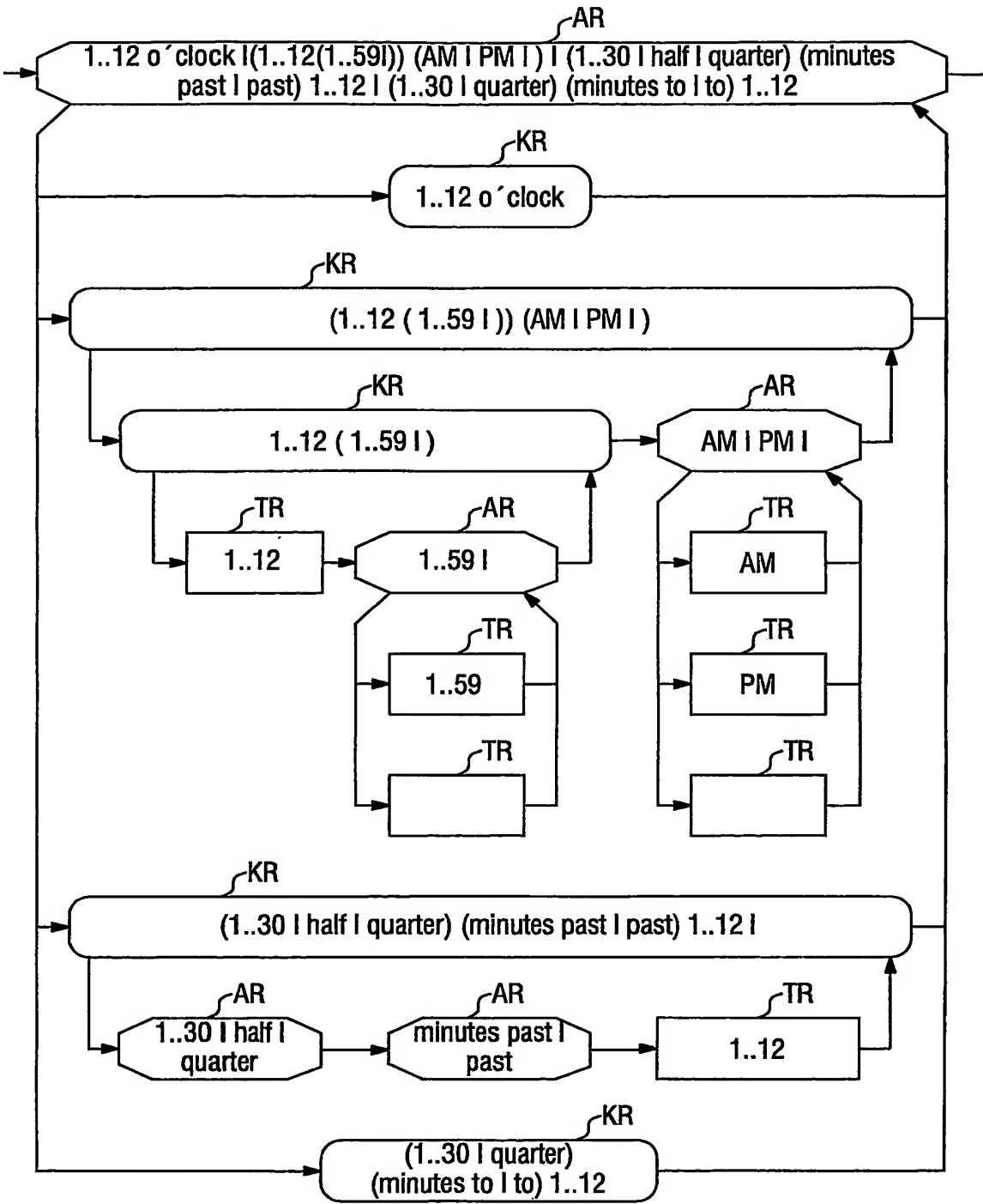


FIG. 6